(12) UK Patent Application (19) GB (11) 2 203 617 (13) A

(43) Application published 19 Oct 1988

(21) Application No 8804685

(22) Date of filing 29 Feb 1988

(30) Priority data
(31) 032185 (32) 30 Mar 1987 (33) US

(71) Applicant
Industrial Technology Institute

(Incorporated in USA-Michigan)

2901 Hubbard, Ann Arbor, Michigan 48106, United States of America

(72) Inventor
Robert Stanley Matthews

(74) Agent and/or Address for Service
J A Kemp & Co
14 South Square, Gray's Inn, London, WC1R 5EU

(51) INT CL⁴
G06F 11/30

(52) Domestic classification (Edition J):
H4P PD
G4A FMG

(56) Documents cited
EP A2 0178473 EP A2 0165603 WO A2 82/03710
US 4477873 US 4437184

(58) Field of search
H4P
G4A
Selected US specifications from IPC sub-class G06F

(54) Embedded test system for communications systems conformance testing

(57) The internet protocol conformance is tested by manipulating the upper layer flow control mechanisms in order to manage lower layer message transmission and reception. The test program controls the credit window by controlling the flow control and acknowledgment mechanisms of the transport protocol to control the internet protocol entity in the system under test. Tests involve monitoring for expected responses in the flow control and/or acknowledgement aspects of message transfer.

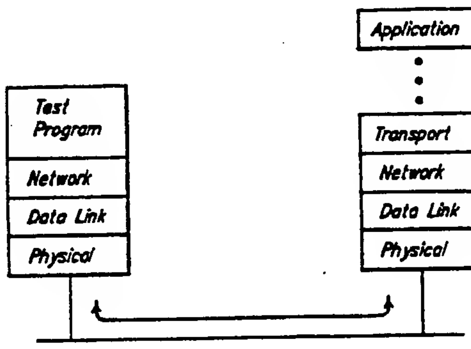


FIG. 6.

GB 2 203 617 A

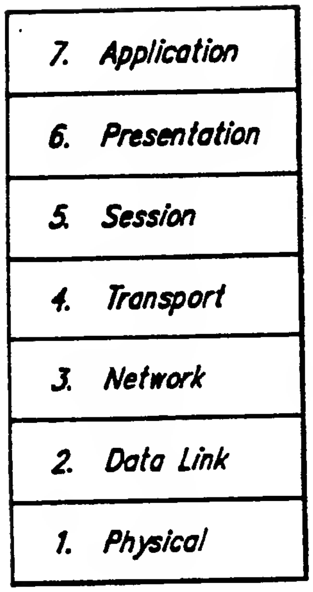


FIG. 1.

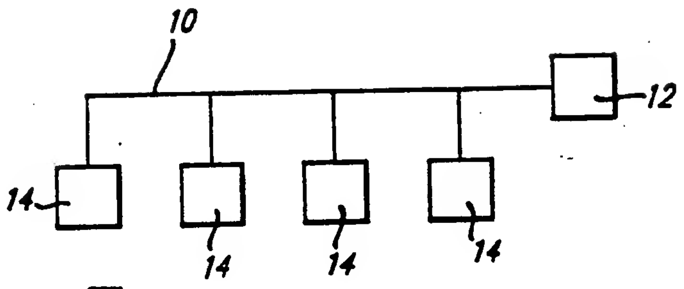


FIG. 2.

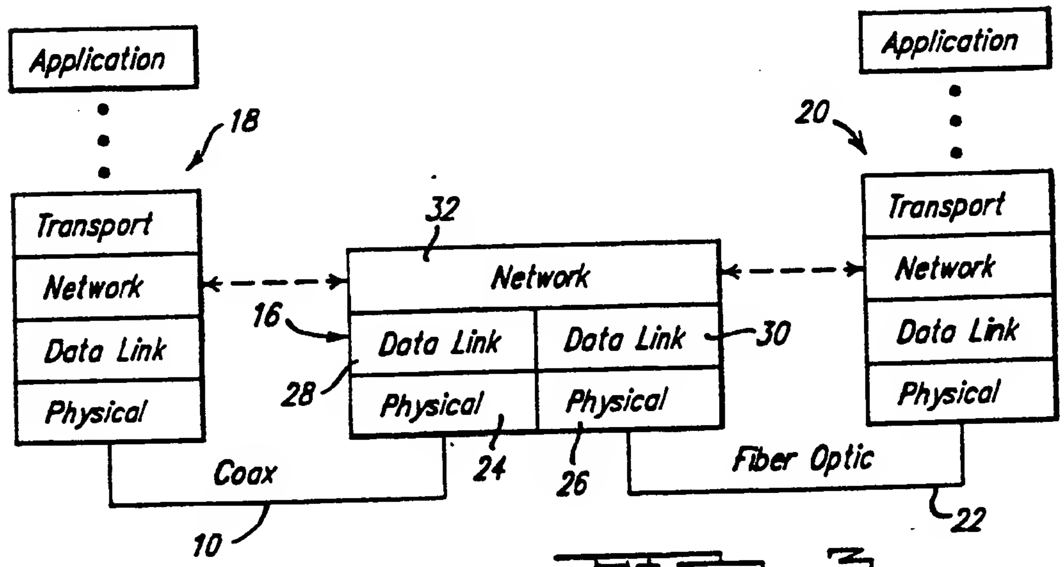


FIG. 3.

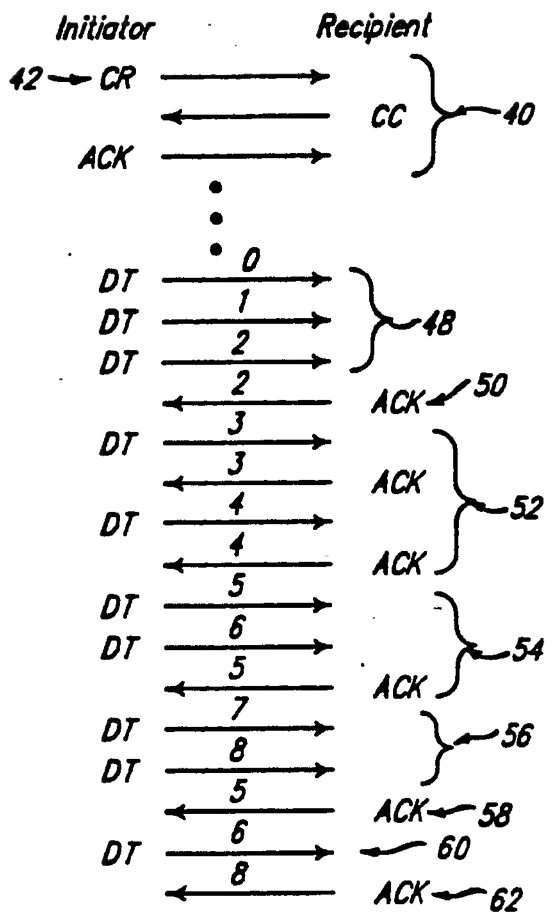

Initiator
Recipient
42
CR
CC
40
ACK
DT 0
DT 1
DT 2
48
2 ACK 50
DT 3
3 ACK
DT 4
4 ACK
52
DT 5
DT 6
5 ACK
54
DT 7
DT 8
56
5 ACK 58
DT 6 60
8 ACK 62

FIG. 4.

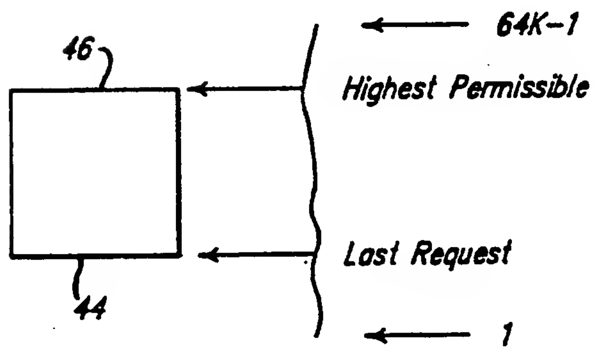

64K-1
46
Highest Permissible
Last Request
44
1

FIG. 5.

2203617

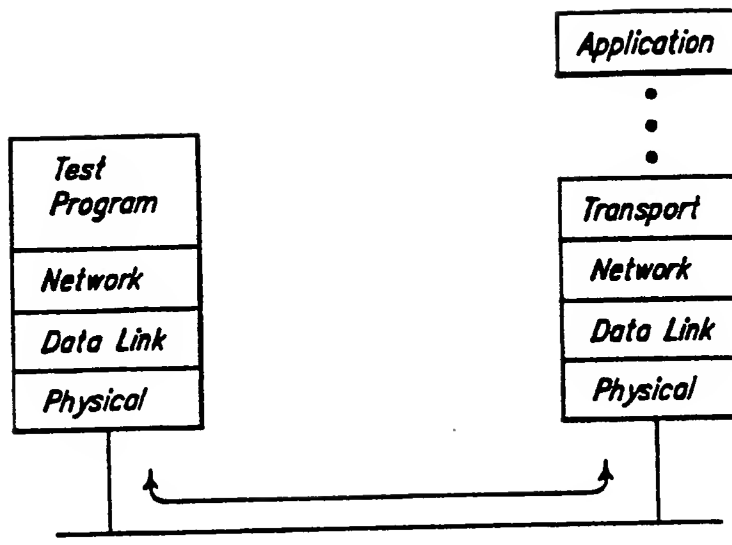


FIG. 6.

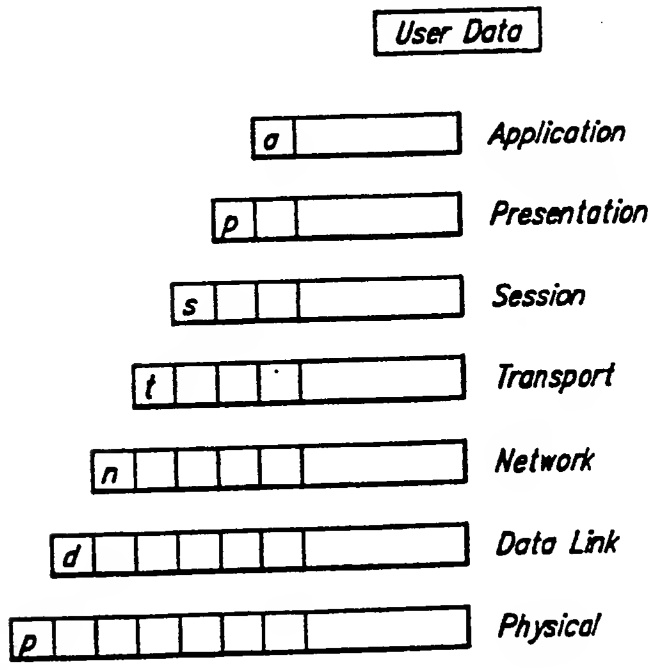
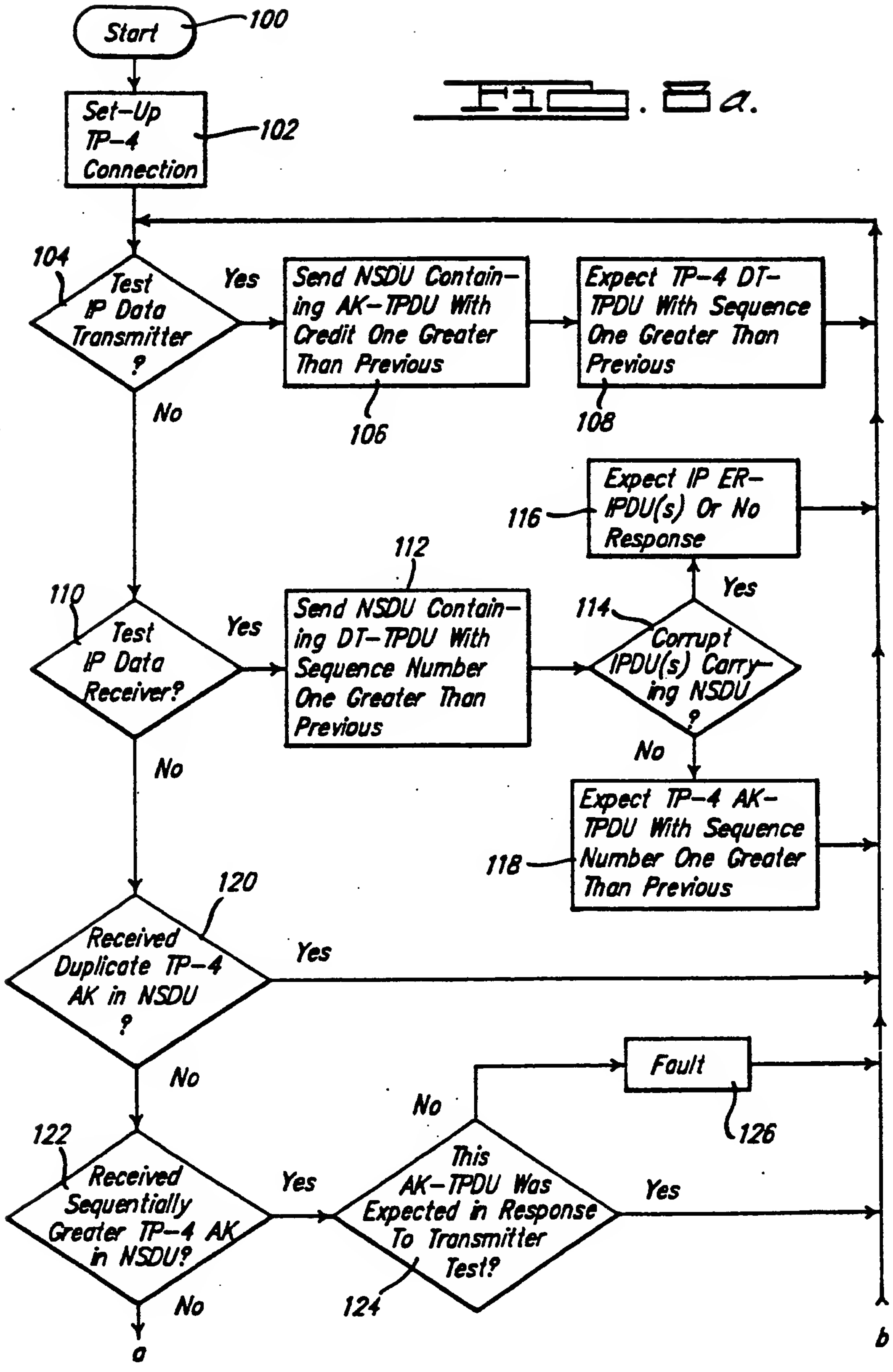


FIG. 7.

FIG. 8a.



Start
100
Set-Up TP-4 Connection
102
104
Test IP Data Transmitter ?
Yes
Send NSDU Containing AK-TPDU With Credit One Greater Than Previous
106
Expect TP-4 DT-TPDU With Sequence One Greater Than Previous
108
No
Expect IP ER-IPDU(s) Or No Response
116
112
110
Test IP Data Receiver?
Yes
Send NSDU Containing DT-TPDU With Sequence Number One Greater Than Previous
114
Corrupt IPDU(s) Carrying NSDU ?
Yes
No
Expect TP-4 AK-TPDU With Sequence Number One Greater Than Previous
118
No
120
Received Duplicate TP-4 AK in NSDU ?
Yes
No
Fault
126
No
122
Received Sequentially Greater TP-4 AK in NSDU?
Yes
This AK-TPDU Was Expected in Response To Transmitter Test?
Yes
124
No
a
b

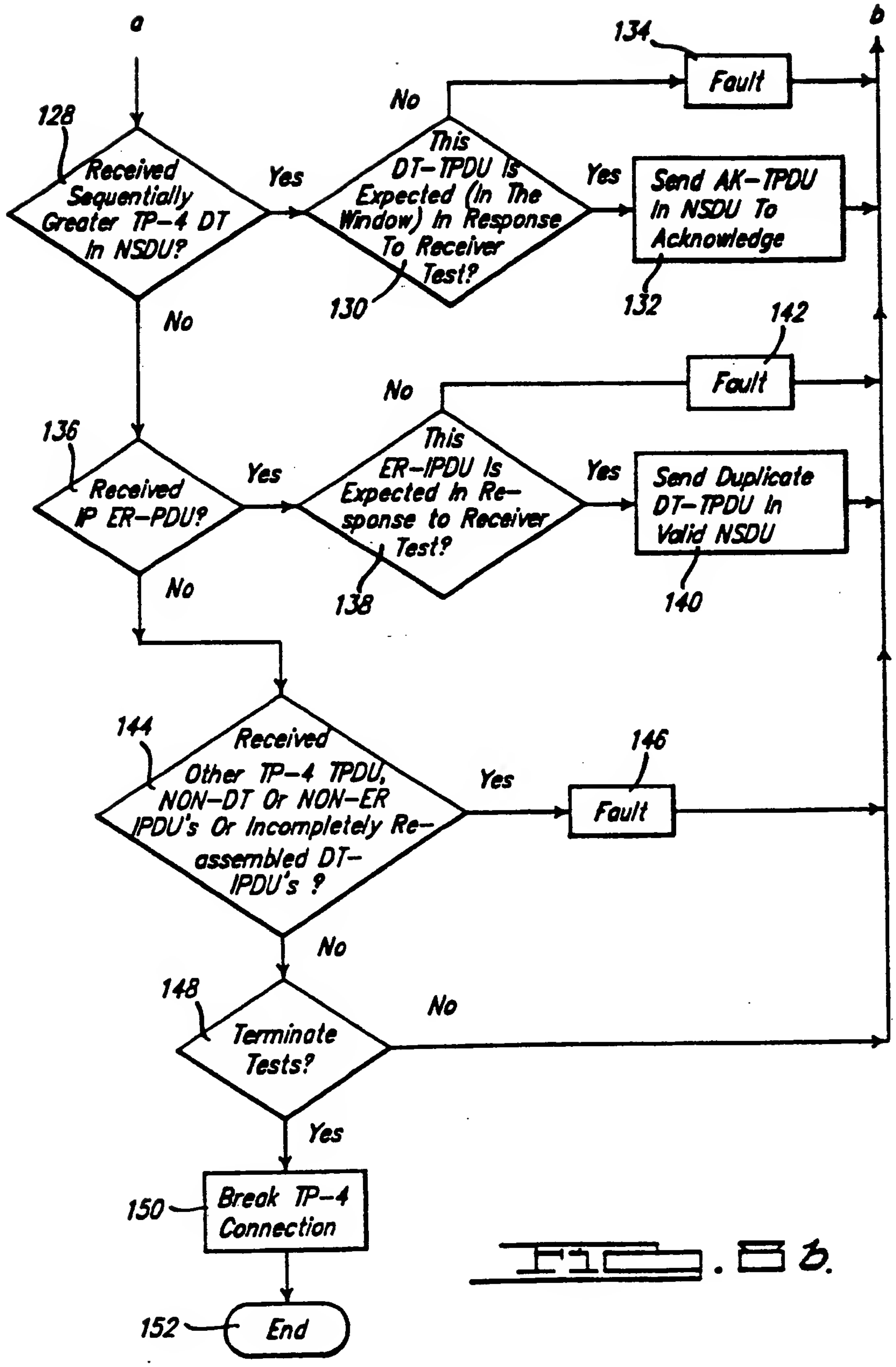

a
b
128
Received Sequentially Greater TP-4 DT In NSDU?
Yes
No
This DT-TPDU Is Expected (In The Window) In Response To Receiver Test?
130
No
134
Fault
Yes
Send AK-TPDU In NSDU To Acknowledge
132
136
Received IP ER-PDU?
Yes
No
This ER-IPDU Is Expected In Response to Receiver Test?
138
No
142
Fault
Yes
Send Duplicate DT-TPDU In Valid NSDU
140
144
Received Other TP-4 TPDU, NON-DT Or NON-ER IPDU's Or Incompletely Re-assembled DT-IPDU's ?
Yes
146
Fault
No
148
Terminate Tests?
No
Yes
150
Break TP-4 Connection
152
End

FIG. 8b.

EMBEDDED TEST SYSTEM FOR COMMUNICATIONS SYSTEMS CONFORMANCE TESTING

2203617

Background and Summary of the Invention

The present invention relates generally to the testing of communications system software. More specifically, the invention relates to a method of determining whether a communications scheme conforms to a specification of communications protocols such as the MAP specification, based on the Open Systems Model established by the International Organization for Standardization (ISO).

Many manufacturing processes today are controlled by networked computer systems which must communicate with one another. In the interest of standardization, the automotive industry, and other manufacturing industries, have proposed a specification of communications protocols which are used to interconnect computer systems and robotic systems to communicate with one another. One such specification is the MAP specification. The MAP specification is based on a model of computer communication established by the ISO called the Open Systems Model. The Open Systems Model (and the MAP specification) divides the communications problem into seven sets of related issues called layers, which are discussed below.

To ensure compatibility with a manufacturer's existing computer and automation equipment, suppliers and vendors of new equipment must develop compatible communications software. The new equipment must conform to the MAP specification.

The MAP specification and the Open Systems Model upon which it is based are comparatively complex communications protocols. Testing a new communications program or a new piece of communicating hardware to determine if it conforms with the MAP specification is not a simple task. As explained more fully below, in order to test all layers of a communications system for conformance it has heretofore been necessary for the new product developer to write and include special test programs directly into the communications software. This adds complexity and expense to the software and can leave the communications system with extra software baggage that is not needed once conformance has been determined. Possibly even more troublesome is the fact that such test routines have heretofore been closely linked to a particular layer within the MAP specification. This close linking has made it very difficult to integrate certain functions between adjacent layers without the test routines getting in the way.

The present invention overcomes this problem by a technique of manipulation of upper layer flow control mechanisms in order to manage lower layer message transmission and reception for conformance testing purposes. The invention allows the new product to be tested without the need to include some of the previously required special test functions in the new product software. The invention permits conformance testing to be effected without interfering between adjacent layers of the software being tested. Thus, the invention allows the communications software to employ integrated or interrelated adjacent layers, such as between the network and transport layers in order to improve software performance.

More specifically, the invention permits the protocol conformance testing to be performed on the network layer of the protocol implementation under test (IUT), that is, the internet protocol entity in the system under test, without altering or appending test code to the network/transport interface (between the network and transport layers) of the product. The invention ndirectly manages message flow between the network/transport interface of the product in order to determine whether the network layer of the implementation under test is passing data to and accepting data from the transport layer correctly. The invention performs the network layer protocol testing in a noninvasive fashion. The invention controls the flow of information through the transport layer, to thereby indirectly control the network service interface through the transport, by indirectly taking control over the credit and acknowledgment signals automatically and periodically generated by the transport layer.

For a more complete understanding of the invention, its objects and advantages, reference may be had to the following specification and to the accompanying drawings.

Brief Description of the Drawings

Figure 1 is a block diagram depicting the seven layered communications system protocol;

Figure 2 is a schematic block diagram illustrating the physical communication topology implemented by the presently preferred communications system;

Figure 3 is a block diagram illustrating an intermediate system or router for transferring messages between nonadjacent nodes;

Figure 4 is a message sequence diagram illustrating a flow of message transmissions and acknowledgments between initiator and recipient;

Figure 5 is a diagram illustrating the window principle used by the transport layer of the communications system to regulate the flow of information;

Figure 6 is a block diagram of the test program of the invention, illustrating the manner in which an exemplary communications system program is tested;

Figure 7 is a diagram illustrating the form of protocol data units at each of the seven levels; and

Figure 8 (8a and 8b) is a flow chart depicting the testing method of the invention.

Description of the Preferred Embodiment

Before a detailed explanation of the invention is given, a brief overview of the MAP specification will be presented. The MAP specification is described in, General Motors, "Manufacturing Automation Protocol Version 2.1," General Motors Corporation 1985. The MAP specification is based upon the Open Systems Interconnection (OSI) Model established by the ISO. A detailed explanation of the OSI Model may be found in ISO IS7498, "Information Systems Processing — Open Systems Interconnection — Basic Reference Model," 1984 which is incorporated herein by reference.

Figure 1 diagrammatically illustrates the seven layers into which the MAP communications protocol is divided. At the lowermost or most primitive level is the first level, the Physical Layer. The Physical Layer is responsible for encoding and physically transmitting messages between two adjacent nodes on the network. The MAP standard uses a bus topology pictured in Figure 2. The Physical Layer implements the IEEE 802.4 standard. Communication is over a coaxial cable 10 which provides two communications channels. One channel is a low frequency channel which propagates in a first direction and the other channel is a high frequency channel which propagates in the opposite direction. A head end unit 12 terminates one end of the cable for shifting the low frequency communications up to the high frequency to thereby change the propagation direction. The communicating units 14 place communications on coaxial cable 10 at the low frequency where they propagate to the head end unit 12 for retransmission at the higher frequency. All communicating units receive data at the high frequency.

Returning to Figure 1, immediately above the Physical Layer is level two, the Data Link Layer. The Data Link Layer is responsible for framing the binary data handled by the Physical Layer into octets (eight bits). A plurality of octets are grouped into data packets by the Data Link Layer. The Data Link Layer connects a link address to each packet which identifies which communicating unit 14 or node is to receive the packet of data. The Data Link Layer is responsible only for addressing data intended for physically connected devices on the coaxial cable. In order to send a message to a nonadjacent node (one which is not physically connected to the coaxial cable) an intermediate systems unit or router is employed.

The third level, or Network Layer, handles the routing of information through intermediate systems or routers. The Network Layer establishes the protocol for transferring messages from a node on one cable to a node on a physically separate cable. Figure 3 depicts an intermediate system or router 16 which interconnects a first communication system 18, which communicates over a coaxial cable 10, with a second communication system 20, which communicates over a fiber optic cable 22. The intermediate system or router 16 employs two Physical Layers 24 and 26 which are compatible with the coaxial cable and fiber optic protocols, respectively. The Data Link yers 28 and 30 respond to the Physical Layers 24 and 26, respectively, and in turn communicate with the Network Layer 32 of the router. In effect, the Network Layer 32 of the router can be considered as a bridge between the respective Network Layers of the first and second communications systems 18 and 20. In this fashion, it is possible to establish communications between two nonadjacent nodes employing similar or dissimilar physical protocols.

In theory, the Network Layer can determine the recipient node of a given communication by a variety of different schemes. One scheme is a connection oriented protocol in which a permanent virtual pipe is created between two devices. Using such a protocol, it is not necessary to place a signature on the message being conveyed, nor is it necessary to place any routing information on the message, as the permanent virtual pipe automatically describes the identity of both the initiator and the recipient. Another protocol is the connectionless oriented protocol. The connectionless oriented protocol requires that a message be enclosed in or accompanied by an address and a signature identifying the desired recipient of the message and also identifying the initiator of the message. The connectionless oriented protocol, unlike the connection oriented protocol, does not guaranty that messages are conveyed in any particular order. The MAP specification presently adopts the connectionless protocol. This connectionless protocol is commonly known as the Internet or IP protocol.

Because the IP protocol does not guaranty that messages are received in the order in which they are sent, some provision must be made to ensure reliable delivery and servicing of messages. Referring back to Figure 1, the fourth level or Transport Layer provides this ssurance that messages are reliably delivered. The MAP specification uses a positive acknowledgment technique in which the recipient of a message acknowledges receipt by returning an acknowledge signal (ACK) to the initiator. The details of this acknowledgment system are explained below.

The Transport Layer also handles message flow control. Flow control is the process used to ensure that the pace of communications is appropriate for the two communicating devices. For example, if communication is to be established between a high speed mainframe computer and a low speed personal computer, certain provisions must be made to establish a working communication rate. This is done by a process of negotiation between the two communicating units, and specifically between the Transport Layers of the two communicating units. The two communicating systems agree in advance of communication on the maximum message (or Protocol Data Unit, PDU) size. As the exchange of messages proceeds, each communicating unit can send control parameters to control the rate of PDU transmission.

The mechanism by which the transmission rate is controlled is based on the PDU sequence number. A unique sequence identification number is attached to each PDU, so that the sequence numbers can be used to place the packets in the proper order should they become misordered. When communication is established, each communicating unit conveys to the other the sequence number corresponding to the maximum number of PDUs which it is prepared to receive. The exchange of data is full duplex, either side can send PDUs to the other up to the maximum sequence number given to it by the other.

As a simple example, if both units are initiating communication and unit A is prepared to receive three PDUs, unit A ld send the number 3 to unit B. Thereafter, if unit A wishes three more PDUs to be sent, it would update the sequence number to 6. Unit B could, at the same time, establish its own desired rate of communication with unit A. Unit B could initially inform unit A that it is prepared to accept PDUs up to and including sequence number 10. After these ten are received, unit B could request another ten, by updating the sequence number to 20, or it could slow the rate down or speed the rate up by selecting either a lower or a higher sequence number. The initial sequence numbers are established during a connection establishment phase of communication. Thereafter, the sequence numbers are updated as part of the message received acknowledgment (ACK) signal which is relayed from recipient to initiator.

To guard against system hang up which would occur if both communicating units were waiting on the other to acknowledge, the Transport Layer entities in each unit are designed or required to routinely send redundant acknowledgment signals which convev the current allowable data messages which can be received. This allowance information is referred to as a window.

For a more complete understanding of the positive acknowledgment system implemented by MAP, refer to Figure 4 which illustrates an exemplary communication between an initiator and a recipient. It should be kept in mind that MAP implements a full duplex communication svstem, both communicating units can be initiators and recipients simultaneously. Referring now to Figure 4, two columns are depicted, designated as initiator and recipient. In these columns are a sequence of message transmissions with arrows indicating the direction of transmission. Communication begins with a onnection establishment sequence. The connection establishment portion of the communication is indicated generally at 40. The sequence begins as at 42 with the initiator sending a connection request (CR) to the recipient. The recipient responds by transmitting a connection confirmation message (CC). The initiator then acknowledges that it received the connection confirmation by sending an acknowledgment (ACK). During this connection establishment phase, both initiator and recipient specify the minimum and maximum sequence number which it is prepared to receive (i.e., the window). For purposes of illustration, only communications initiated by first unit and received by a second unit are illustrated. It will be understood that a similar diagram would result if the communications initiated by the second unit for reception by the first unit were to be illustrated.

When the connection is first established, communication is assumed to begin with sequence number 0 (the first message of transport user information). The highest permissible sequence number established by the recipient, together with the initial sequence number 0 establishes the number of messages which can be sent to the recipient. This number may be conveniently represented as a window illustrated in Figure 5 bounded between the last sequence request (in this case the initial sequence 0) and the highest permissible sequence number requested. If, for example, the recipient wishes to limit the reception to 100 messages, then the lower edge 44 of the window would initially rest at sequence number 0 while the upper edge 46 would be at sequence 100.

Returning to Figure 4, once the connection establishment 40 has occurred, the initiator commences data transmission PDUs (DT).

.r purposes of illustration, it will be assumed that the recipient has requested three messages (PDUs) to be sent. Accordingly, the initiator transmits the first three messages corresponding to sequence numbers 0, 1 and 2. This is indicated generally at 48. When the recipient receives the third sequence, it sends an acknowledgment signal (ACK) back to the initiator. This is indicated generally at 50. The acknowledgment signal contains the sequence number of the highest sequence received, in this case message 2. Thus the initiator knows that the recipient has received all three packets. If desired, the recipient could, with its acknowledgment, change its window size by requesting an additional group of packets different from the number originally requested. For example, the recipient may wish to allow the transmission of up to ten messages. In this case the recipient would revise its highest permissible sequence number to 13. If the recipient does not revise the highest permissible sequence number, then it remains at the earlier selected value, 3, and no further sequences can be transmitted.

Assuming that more sequences have been requested, the initiator would continue to send messages, and would expect to receive acknowledgments thereafter. This is shown for messages 3 and 4 as at 52. It is, of course, possible that a particular message mav not be received in the proper sequence. A given packet can become temporarilv delayed or lost during transmission. At 54, the initiator sends messages 5 and 6. The recipient acknowledges only message number 5 to illustrate the case in which message number 6 is not sequentially received. Meanwhile, the initiator continues to send messages 7 and 8, as at 56. The recipient, however, cannot acknowledge receipt of message numbers 6, 7 and 8, since it has not yet acknowledged message number 5.

In accordance with the standard operation of the Transport Layer protocol, the recipient is required to periodically send an acknowledgment reflecting the last sequence number which it has received. In this case, the recipient has received message number 5. Accordingly, at step 58 the recipient sends a periodic acknowledgment of receipt of message number 5. The initiator determines from this acknowledgment that the recipient has not yet received message number 6 (which was sent earlier) at 54. Accordingly, the initiator resends message number 6, at 60 and the recipient then responds with an acknowledgment of message number 8. In this example, it is presumed that only message number 6 was not received by the recipient and that the message numbers 7 and 8 were received and were stored pending receipt of message number 5.

It is important in understanding the invention to recognize that the window defined by the upper and lower edges illustrated in Figure 5 can change in size with each acknowledgment. The lower window edge 44 represents the last sequence number acknowledged and the upper window edge represents the maximum sequence number permitted. The present invention is primarily concerned with controlling data flow between the level three Network Layer entities through use of level four Transport Layer mechanisms. For completeness, a brief description of the level five, six and seven layers will be presented.

The level five or Session Layer illustrated in Figure 1 is primarily involved in synchronizing the communicating units to take turns in carrying out a communication. The Session Layer is roughly alogous to sentence punctuation, which breaks one complete thought, allowing another message, possibly from the other unit to reply.

The sixth level or Presentation Layer is responsible for handling the data representation between communicating units. Communicating units may not communicate using similar encoded languages or similar file types. The Presentation Layer provides an agreed upon common language, encoding or file type by which the two incompatible systems may communicate. The Presentation Layer can also perform data encoding and decoding for data encryption and the like. Although the OSI Model permits communicating systems to negotiate protocol at this level, the presently implemented MAP specification does not employ such negotiation.

The final and seventh level is the Application Layer. The Application Layer defines for certain contexts what language is to be used during communication. The context can be either implicit, based upon the nature of the communication or it can explicitly stated. The Application Layer is needed because data being communicated may be organized in fundamentally different ways by different computer systems. The Application Layer defines a mutually agreed upon way in order to refer to things. One or both of the communicating units may have to translate its data to the mutually agreed upon format if such format is not inherently provided.

As stated above, the present invention is involved primarily with testing the conformance of the level three Network Layer internet protocol entity of a system under test. The objective is to transmit both valid and invalid data to the Network Layer entity in order to determine whether it properly distinguishes between the two. Another objective is to cause the Network Layer to pass data up to the

Transport Layer and also to receive data passed down from the Transport Layer without error. The Network Layer is connectionless in that it does not implement connection-oriented protocols as some of the upper layers do. Prior to this invention, in order to test the Network Layer, it has been necessary to place a special test program in the Transport Layer of the system under test (or at the Network/Transport Layer interface) for the purpose of sending predefined test messages through the transport/network interface to the Network Laver in order to see if the Network Layer properly receives the same. The test program also must respond to predefined test messages sent through the transport/network interface from the Network Layer in order to determine whether the Network Layer properlv convevs messages. The addition of this test software adds to the complexity of the communication system and makes it difficult to design high performance software which may need to integrate certain closely related functions of the Transport and Network Layers.

Having thus presented a general overview of the invention and the presently preferred operating environment, a more detailed explanation of the embedded test system will now be given.

The invention is referred to as the "embedded" test system or technique because the network service interface in the system under test is allowed to remain naturally coupled to the Transport Layer. The common test method in which the network service interface is required to be decoupled from the Transport Layer and driven bv specialized software in the system under test is referred to as the "exposed" test system or technique.

The invention utilizes as its implementation base the software which was developed to perform the exposed test technique.

Furthermore, the invention is capable of executing the same set of test cases as the exposed test system. The choice of technique is left to the designer of the system under test. In other words, the "embedded" test system uses the existing "exposed" internet test system software as its implementation base. In fact, only one test system is maintained, with a user command to select whether an embedded or an exposed service control is available.

The exposed version continues to use the NBS defined Test Management Protocol Data Units (TMPDU) and the associated Upper Tester for control of the IUT service interface. A description of the TM (test management) Protocol can be found in ICST/SNA-85-7, National Bureau of Standards, "The Design of a Test System for Implementations of ISO Connectionless Network Protocol," July 1985 (NBS 86).

Figure 8 illustrates the presently preferred method of implementing the embedded test system employing the invention. The sequence starts at step 100 and proceeds to set up the connection oriented communication between the test system software and the system under test at 102. The remainder of the test sequence illustrated in the flow chart of Figure 8 comprises a plurality of branch points (at at 104, 110, 120, 122, 128, 136 and 144) where specific tests can be performed.

For example, if it is desired to test the internet protocol (IP) data transmitter, flow branches at 104 to execute steps 106 and 108. In step 106 a predefined unit of data, referred to as an NSDU, containing an AK_TPDU is sent with a credit one greater than the previous sequence number. This opens the credit window to allow one message unit to be returned from the system under test. In implementing these tests the system under test would normally be preconditioned to send and expect to receive a large number of octets of data. By opening the credit window with a credit one greater than the previous sequence, one of the plurality of data octets is permitted to flow from the system under test to the test system. At step 108, the test system sets a flag to expect to receive a DT_TPDU with a sequence one greater than the previous.

If it is desired to test the IP data receiver, step 110 branches control to step 112 which sends an NSDU containing a DT_TPDU with a sequence number one greater than the previous. A decision has been made at step 114 whether to corrupt the IPDUs which carry the NSDU. If corrupted, a flag is set to tell the test system to expect an error message ER_IPDU or no response. If the data is not corrupted, a flag is set to tell the test system to expect an acknowledgment message AK_TPDU with a sequence number one greater than the previous.

As explained above, the map communications protocol requires periodic sending of duplicate acknowledgment signals to guard against system hang up. These duplicate acknowledgment signals are intercepted at branch point 120. If a duplicate acknowledgment signal is received, no action is taken by the test system and flow control simply returns to the top of the loop.

In order to test whether the acknowledgment sequence flagged in step 18 is received, step 122 monitors receipt of the expected acknowledgment and branches to step 124 if the acknowledgment is received. Step 124 tests the acknowledgment to see if it was in response to the transmitter test and if so no fault is reported, otherwise a fault is reported at 126.

A similar test is performed at step 128 to determine if the DT_TPDU flagged at step 108 has been received. If so, step 130 tests to determine whether this was in response to a receiver test. If so, an acknowledgment signal AK_TPDU is sent at 132. Otherwise a fault condition is signaled at 134.

In a similar fashion at step 136 a test is performed to determine whether the expected error message flagged in step 116 is received. If so, a determination is made as to whether this error signal ER_IPDU is in response to a receiver test. If so, a duplicate DT_TPDU is sent. Otherwise a fault condition is signaled at 142.

Finally, if any other non-DT or non-ER IPDUs are received or if any incompletely reassembled DT_IPDUs are received control branches at 144 to signal a fault at 146. At 148 a decision is made whether to terminate the testing or whether to return to the top of the loop for further testing. If testing terminates, the connection is broken at 150 whereupon the test program ends at 152.

The embedded version uses standard Transport Protocol Data Units (TPDU) and a MAP 2.1 conforming Transport Entity for IUT network service interface control. The IUT Transport Entity is, in turn, controlled by the Remote Scenario Interpreter as used in Transport testing. Only one transport scenario (test case) is required for use in embedded IP testing. Thus no direct access to the IUT's network service interface is required.

Controlled use of MAP/ISO Class 4 Transport Protocol in embedded testing provides the same functionality for controlling and observing Network Service interface activity as the custom protocol required in the exposed method. (There are minor exceptions, but the lost TM functionality is unnecessary to begin with.) During internet otocol (IP) Test Case execution, only DT_TPDUs and AK_TPDUs are carried in the IPDU data portions. Transport connection establishment and termination are carried on outside the execution of any IP Test Case.

Connection establishment is initiated through issuance of a "connect" command. A CR-CC-AK TPDU exchanged is expected. The test system continues to transmit CRs for a reasonable time before giving up. IP Test Cases only execute if a connection is deemed open.

Transport timer and parameter values are established through CR-CC negotiation, or predefined in the embedded IP Test Plan. These are expected to be similar to values used in current transport protocol conformance tests.

Valid IPDUs data sent by the Lower Tester are correctly sequenced DT_TPDUs with data format observing the Remote Scenario Interpreter expectations. The Remote Scenario Interpreter (RSI) is the same as used in standard transport protocol testing. IPDU data sent in invalid IPDUs is also sent as correctly sequenced DT_TPDUs. If the invalid IPDU is accepted, then the test system signals a fault. If a correct error response is returned, then the testing continues as passing. In order to maintain correct transport message sequencing, the DT_TPDU will be retransmitted in a valid IPDU after the correct error response is received.

An AK_TPDU with correctly increased lower window edge is expected in response to a DT_TPDU carried in a valid IPDU.

In order to restrict the IUT Transport Entity from prematurely sending DT_TPDUs, AK_TPDUs sent by the test system supply no credit. When an NSDU (Network Service Data Unit, i.e., a correctly sequenced DT_TPDU) from the IUT is desired, an AK_TPDU granting one edit is sent by the Lower Tester. If the Lower Tester receives a valid NSDU (i.e., a valid DT_TPDU) the window is reclosed and the DT_TPDU acknowledged. This AK_TPDU granting credit does not specify NSDU size. Such a specification seems to be undesirable, since it would attempt to control an implementation specific parameter. Thus for conformance testing purposes, the AK with credit is sufficient. Note that the DT_TPDU data from the IUT is expected to follow correct Remote Scenario Interpreter format.

Both the Lower Tester and the IUT are required to periodically transmit an AK_TPDU to satisfy the Transport window time requirement. This enables both parties to be cognizant of each other's continued viability.

Embedded Service Control and Observation:

The exposed mode system uses a conformance entry to keep track of all expected IPDU from the IUT. This table is appropriately filled upon the test system sending an IPDU, and the table entry is removed upon reception of satisfying IPDU.

In embedded mode, this table is not used in connect and disconnect processes. For connect/disconnect processes only the transport connection context data structure is consulted.

During IP test case execution, both the conformance table entries and transport context is used. Since only DT_TPDUs and AK_TPDUs are used during IP test case execution, this is straightforward. The test system uses the following rules:

1. DT_TPDU is received: if the DT is in sequence, it is in the open window, the data portion is correctly formatted, and a conformance entry generated by an AK_TPDU with credit has been

sent, then the entry is resolved, the "Tctx" is updated, and an AK_TPDU is sent by the Lower Tester.

2. AK_TPDU with duplicate seq# is received: update the "Tctx" in an appropriate manner.

3. AK_TPDU with greater seq# is received: if a "Valid Data IPDU"-type conformance entry exists and an outstanding DT_TPDU has been transmitted, then the entry is resolved, and the "Tctx" is updated.

4. DT_TPDU within valid data IPDU is sent: a good DT_IPDU is to be sent of size n octets of data according to the Test Case. A "Valid Data IPDU"-type conformance entry is generated, and the IPDU is sent.

5. DT_TPDU within invalid data IPDU is sent: a bad DT_IPDU is to be sent carrying n octets of data, and error report flag is set on, as according to the Test Case. A "Valid Data IPDU"-type conformance entry is generated, and the IPDU is sent.

6. AK_TPDU with duplicate seq# is sent: the "Tctx" indicates that the window timer has expired.

7. AK_TPDU with duplicate seq# and credit is sent: an NSDU carrying a correctly sequenced DT_TPDU is expected. A conformance entry is made and the data IPDU with AK_TPDU is sent.

8. AK_TPDU with greater seq# is sent: a "Valid Data IPDU" entry has been resolved and a data IPDU is sent.

9. ER_IPDU is received: if "Error-IPDU" conformance entry exists, then the entry is resolved.

Integration with Current Internet Software:

The embedded test functionality is integrated with the existing software in manner such that the changes are isolated and clearly structured. An overview of the exposed system software structure is presented, followed by the changes made to support the embedded system.

The exposed system software is structured as a continually looping process. The modifications/alterations to support the embedded system work within this structure. The following pseudo-code describes the dual-mode system. Statements which are additions are marked "*A"; modifications are marked "*M."

```
forever
[
     if   (command from user terminal)
*M        execute command;
*A   if   (embedded mode)
*A
*A        send connection maintenance TPDUs;
*A        send AKs to confirm valid DTs;
     if   (test case has been queued && no test case in-progress)
          read test case;
*M        build abstract IPDUs'
          queue abstract IPDUs for transformation;
*M        set up conformance entries for receiving;
          test case is considered in-progress;
     if   (abstract IPDUs are queued)
          transform abstract IPDUs into concrete IPDUs'
          queue concrete IPDUs for sending;
```

C C

```
    if   (concrete IPDUs are queued)
         send concrete IPDUs;
    if   (incoming IPDUs are in receive queue)
         validate IPDU header;
*M       check data and reassemble;
*M       evaluate conformance to test case;
    if   (test case timer event)
         take appropriate action;
    if   (test in progress && all conformance entries resolved)
         test is finished;
}
```

Modifications to support the embedded system are as follows:

1. The global data structure "Tctx" provides necessary data to support operation of the Transport connection. This includes the variables needed to generate and analyze data for the Remote Scenario Interpreter.
2. The IPDU data generated at abstract IPDU formation time is provided by a function named gen_data(). This function is modified to generate appropriate DT_TPDUs in embedded mode. Minor modifications to the conformance entry-making logic are required.
3. IPDUs are generated to request the IUT to send data. The data portion of such IPDUs carries "Send-NSDU" TMPDUs in exposed mode. These TMPDUs are generated at abstract IPDU formation time by the function "nsdu_data()." This function is modified for embedded mode to generate AK_TPDUs with credit(s). Minor modifications to the conformance entry-making logic are required.

Received IPDUs have their data portions analyzed by several different routines in exposed mode. These routines switched to depending upon whether the data was identified to be a TMPDU, or just sequenced data. These routines are called from either the "receiver()" or "reassembler()" function.

5. In embedded mode, only a single routine named "scan_data()" is called. This routine is called from within "reassembler()." All IPDU data portions are passed through to "reassembler()" whether reassembly is required or not (this is the existing practice for exposed mode also). The "scan_data()" routine is modified to apply transport PDU type analysis. Received DTs are expected to be within the allowable window. Received AKs are expected to acknowledge only outstanding DTs, or be duplicates for the purposes of connection maintenance. The "scan_data()" function returns a count of errors detected and "reassembler()" prints a message indicating either encoding or length violations.

6. Transport connection is managed by the routine "transport()." This routine is called in the forever loop after the user command processing has been called. User connect, disconnect, and close commands (if valid) set "Tctx" variables indicating that a desired service be achieved. The "transport()" routine sends any required concrete IPDUs directly. Received IPDUs are handled by "scan_data()" within "reassembler()." The "scan_data()" bases its checking upon the indicators in "Tctx."

Conformance Analysis:

The existing conformance analysis mechanism is used nearly unmodified for embedded mode. The changes are (1) the sequence number variable in each conformance entry is adjusted to correlate with the transport PDU activity; and (2) an entry may require more than one response to be resolved. This implies that any existing bugs to the conformance analysis mechanism still have not been corrected.

Transport level errors are announced by error messages to the operator console. It is the operator who must decide whether such an error message indicates inconsequential activity, non-testability, or conformance failure.

Transport Connection Usage:

The transport connection is always initiated by the test system. The LT proposes low-bids on most negotiations, giving the IUT no choice. These bids are: checksums on, normal format, and no expedited data. The maximum TPDU size is proposed to be 512 octets.

This size value is chosen to support the current hardware limitations of the test system. The IUT may negotiate this down. Regardless of the negotiated maximum size, the test system never sends an IPDU greater than the maximum message size in the test system (currently 600 octets). The current set of IP Test Cases does not require that an NSDU larger than 512 octets be sent or received by the test system. The test system also initiates the disconnect when commanded by the console.

Every transmitted AK carries the flow control-confirmation (fcc) parameter. Reception of naked (parameterless) AKs result in a retransmission of the latest AK.

Date Structures:

The following data structure is used to maintain the transport context:

```
ruct TCONTEXT Tctx

{

int state;              /* major state */
int TCref;              /* LT reference number */
int RTref;              /* IUT reference number */
int sizebid;            /* maximum negotiated tpdu size */
int send_seq;           /* sending sequence number */
int send_ssq;           /* sending subsequence number */
int send_cdt;           /* sending credit */
int recv_seq;           /* receiving sequence number */
int recv_ssq;           /* receiving subsequence number */
int recv_cdt;           /* receiving credit */
int window_time;        /* window timer value */
int inact_time;         /* inactivity timer value */
int giveup_time;        /* giveup timer value */
int retran_time;        /* retransmission timer value */
int retran_count;       /* retransmission count */
int cr_len;             /* cr pdu length */
int ak_len;             /* ak pdu length */
int dt_hlen;            /* dt pdu header length */
int dt_tlen;            /* dt pdu total length */
int dr_len;             /* dr pdu length */
int dc_len;             /* dc pdu length */
int ip_hlen;            /* ipdu header length */
int ip_dui;             /* next dui to use */
unsigned
 char crpdu[512];       /* cr tpdu for sending */
unsigned
 char akpdu[512];       /* ak tpdu for sending */
unsigned
 char dtpdu[512];       /* dt tpdu for sending */
unsigned
 char drpdu[512];       /* dr tpdu for sending */
unsigned
 char dcpdu[512];       /* dc tpdu for sending */
unsigned
 char ippdu[700];       /* data ipdu for sending */
 long send_count;       /* count of transport data sent */
 long send_goal;        /* total amount of data to be sent */
 long recv_count;       /* count of transport data received */
 long recv_goal;        /* total amount of data to be received */
 int akstat;            /* classification of ak type */
 int send_dtseq;        /* sequence number of last dt sent */
 int send_akseq;        /* sequence number of last ak sent */
 int send_eot;          /* eot sent */
 int recv_eot;          /* eot received */
 int timebase;          /* time at program initialization */
 int send_time;         /* time last data transmitted */
 int recv_time;         /* time last data received */
 int send_vector;       /* vector indicating next pdu to send */
 int reason;            /* disconnect reason */
 int dest;              /* out-of-context pdu dest */
int src;            /* out-of-context pdu src */
int TCsent roll;    /* count of dt sent sequence number rollover */
int TCack roll;     /* count of ak sent sequence number rollover */
int RTsent roll;    /* count of dt recv sequence number rollover */
int RTack roll;     /* count of ak recv sequence number rollover */
char *fault;        /* pointer to pdu error */
struct
 DTREG DTreg[10];   /* registers to buffer sent, but unacked dts */
struct
 INPDU *pduptr;     /* pointer to received pdu */
```

Connection Maintenance:

Transport TPDUs are formed for sending for two purposes. They are either required to form the IPDU data portion defined by an IP Test Case, or they are required to set up, maintain or terminate a transport connection.

The TPDUs used for IP Test Case support are generated at abstract IPDU formation time. Construction of these TPDUs (only DTs or AKs) reference the "Tctx" structure for appropriate values.

TPDUs sent for connection maintenance are sent directly (the normal sending queues are bypassed). These TPDUs are sent after console command processing. The IPDU header is predefined with the DUI (data unit identifier), segment length indicator, total length indicator and header checksum appropriately modified. All this information is found in the "Tctx." DUI values start at (decimal) 1,000 so as not to conflict with IP Test Case DUIs. IP Test Cases must refrain from specifying IPDUs with DUI values greater than 1,000 to avoid potential conflict. Since all current IP Test Cases use DUIs beginning at (decimal) 1 (or for inactive subset tests no DUI at all). IPDUs used for control are easily distinguishable from IPDUs under test.

All received IPDUs (both modes) are accessed through the "scan_data()" function. In exposed mode, the IPDU and its data are evaluated during reassembly, and a data structure named "segmt," of type "REASM_SEG," is created to maintain pertinent information about the segment. The buffer holding the actual segment is then released. The reassembly process maintains a linked list of "segmt" structures for use in reassembly.

The data type "REASM_SEG" is modified to support embedded mode by maintaining additional information relevant to TPDUs. This includes partially reassembled IPDUs. If the IPDU data is valid according to transport protocol rules and Scenario Interpreter goals, then only the conformance entry need be satisfied. This entry is satisfied if the sequence number of the incoming DT or AK correlates to that of the conformance entry.

Duplicate incoming AKs simply reset the inactivity time and are discarded. If the duplicate is "naked," a flag is set in "Tctx" indicating that an AK is to be transmitted. All transmitted AKs contain the fcc parameter. All other out-of-sequence DTs and AKs are identified as errors.

Transmission of TPDU is triggered by flags being set in "Tctx." In the main loop, immediately after checking for console commands, the function "transport()" is called. If flags are set, then an appropriate IPDU is sent. The mechanism is used for sending all TPDUs except for those DTs and AKs which are required as IPDU data during an IP Test Case.

Template TPDUs are maintained in the "Tctx" structure. Such a TPDU is copied into a buffer containing an IPDU header, and the appropriate parameters are modified in the IPDU header and the TPDU.

~his buffer is then passed to the device write routine, bypassing the queue structure used for IPDUs generated by the IP Test Cases.

The "transport()" routine sends TPDUs needed to support a user command (connect, close, disconnect), AK a valid DT and close a window (which previously was permitted by an IP Test Case generated AK), and to satisfy standard transport timers.

CLAIMS

1. A method of testing the conformance of an implementation of an internet protocol within a system under test to its specification, comprising:

using the flow control and acknowledgment mechanisms of transport protocol to control the internet (IP) protocol entity in the system under test.

2. The method of Claim 1 wherein said flow control and acknowledgment mechanisms are used to maintain a controlled test environment.

3. The method of Claim 1 wherein said flow control mechanisms specify the amount of messages expected to be transmitted by the internet protocol entity in the system under test.

4. The method of Claim 1 wherein said acknowledgment mechanism is used to determine whether the internet protocol entity in the system under test correctly receives and delivers data to a higher user level in the system under test.

5. The method of Claim 1 wherein said flow control mechanism is used to define a window for allowable transmissions in the internet protocol entity in the system under test.

6. A method of testing the conformance of an implementation of an internet protocol within a system under test to its specification, comprising:

using the flow control mechanism of transport protocol to control the internet (IP) protocol entity in the system under test.

7. A method of testing the conformance of an implementation of an internet protocol within a system under test to its specification, comprising:

using the acknowledgment mechanism of transport protocol to control the internet (IP) protocol entity in the system under test.

8. A method of testing the conformance of an implementation of an internet protocol, such method being substantially as hereinbefore described with reference to the accompanying drawings.

Published 1988 at The Patent Office, State House, 66/71 High Holborn, London WC1R 4TP. Further copies may be obtained from The Patent Office, Sales Branch, St Mary Cray, Orpington, Kent BR5 3RD. Printed by Multiplex techniques ltd, St Mary Cray, Kent. Con. 1/87.